Abyan Ardiatama

Praktikum Pembelajaran Mesin

Pertemuan 6 Pengenalan Deep Learning

```
import numpy as np
import tensorflow as tf
import keras
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Hello World terhadap Neural Networks

```
#1. hello world
model = keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

xs = np.array([0, 1, 2, 3, 4, 5], dtype=float)
ys = np.array([1, 3, 5, 7, 9,11], dtype=float)

model.fit(xs, ys, epochs=500)
print(model.predict([10.0]))
```

```
    1/1 [==============================] - 0s 39ms/step - loss: 6.3214e-05
    Epoch 473/500
    1/1 [==============================] - 0s 33ms/step - loss: 6.2473e-05
    Epoch 474/500
    1/1 [==============================] - 0s 19ms/step - loss: 6.1733e-05
    Epoch 475/500
    1/1 [==============================] - 0s 55ms/step - loss: 6.1010e-05
    Epoch 476/500
    1/1 [==============================] - 0s 45ms/step - loss: 6.0288e-05
    Epoch 477/500
    1/1 [==============================] - 0s 9ms/step - loss: 5.9579e-05
    Epoch 478/500
    1/1 [==============================] - 0s 11ms/step - loss: 5.8877e-05
    Epoch 479/500
    1/1 [==============================] - 0s 12ms/step - loss: 5.8181e-05
    Epoch 480/500
    1/1 [==============================] - 0s 10ms/step - loss: 5.7498e-05
```

```
Epoch 481/500
1/1 [==============================] - 0s 8ms/step - loss: 5.6818e-05
Epoch 482/500
1/1 [==============================] - 0s 13ms/step - loss: 5.6150e-05
Epoch 483/500
1/1 [==============================] - 0s 17ms/step - loss: 5.5489e-05
Epoch 484/500
1/1 [==============================] - 0s 11ms/step - loss: 5.4836e-05
Epoch 485/500
1/1 [==============================] - 0s 13ms/step - loss: 5.4189e-05
Epoch 486/500
1/1 [==============================] - 0s 11ms/step - loss: 5.3550e-05
Epoch 487/500
1/1 [==============================] - 0s 11ms/step - loss: 5.2918e-05
Epoch 488/500
1/1 [==============================] - 0s 10ms/step - loss: 5.2296e-05
Epoch 489/500
1/1 [==============================] - 0s 10ms/step - loss: 5.1678e-05
Epoch 490/500
1/1 [==============================] - 0s 11ms/step - loss: 5.1069e-05
Epoch 491/500
1/1 [==============================] - 0s 10ms/step - loss: 5.0469e-05
Epoch 492/500
1/1 [==============================] - 0s 11ms/step - loss: 4.9874e-05
Epoch 493/500
1/1 [==============================] - 0s 12ms/step - loss: 4.9287e-05
Epoch 494/500
1/1 [==============================] - 0s 14ms/step - loss: 4.8705e-05
Epoch 495/500
1/1 [==============================] - 0s 11ms/step - loss: 4.8132e-05
Epoch 496/500
1/1 [==============================] - 0s 13ms/step - loss: 4.7565e-05
Epoch 497/500
1/1 [==============================] - 0s 15ms/step - loss: 4.7004e-05
Epoch 498/500
1/1 [==============================] - 0s 11ms/step - loss: 4.6450e-05
Epoch 499/500
1/1 [==============================] - 0s 10ms/step - loss: 4.5901e-05
Epoch 500/500
1/1 [==============================] - 0s 12ms/step - loss: 4.5362e-05
1/1 [==============================] - 0s 143ms/step
[[21.021557]]
```

Mendeklarasikan objek tipe MNIST dan memuatnya dari database keras:

```
fashion_mnist = tf.keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_da
train_images = train_images / 255.0
test_images = test_images / 255.0
```

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dense(10, activation=tf.nn.softmax)
])
```

```
#mengcompile model
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=T
              metrics=['accuracy'])

#melatih model
model.fit(train_images, train_labels, epochs=10)

#menguji akurasi model
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
print('\nTest accuracy:', test_acc)

#membuat prediksi
probability_model = tf.keras.Sequential([model,
                                         tf.keras.layers.Softmax()])
predictions = probability_model.predict(test_images)
```

```
Epoch 1/10
1875/1875 [==============================] - 7s 3ms/step - loss: 1.7052 - a
Epoch 2/10
1875/1875 [==============================] - 8s 4ms/step - loss: 1.6216 - a
Epoch 3/10
1875/1875 [==============================] - 11s 6ms/step - loss: 1.6097 -
Epoch 4/10
1875/1875 [==============================] - 6s 3ms/step - loss: 1.6034 - a
Epoch 5/10
1875/1875 [==============================] - 6s 3ms/step - loss: 1.5958 - a
Epoch 6/10
1875/1875 [==============================] - 6s 3ms/step - loss: 1.5931 - a
Epoch 7/10
1875/1875 [==============================] - 6s 3ms/step - loss: 1.5885 - a
Epoch 8/10
1875/1875 [==============================] - 6s 3ms/step - loss: 1.5839 - a
Epoch 9/10
1875/1875 [==============================] - 6s 3ms/step - loss: 1.5821 - a
Epoch 10/10
1875/1875 [==============================] - 6s 3ms/step - loss: 1.5796 - a
313/313 - 43s - loss: 1.5981 - accuracy: 0.8627 - 43s/epoch - 136ms/step

Test accuracy: 0.8626999855041504
313/313 [==============================] - 1s 3ms/step
```

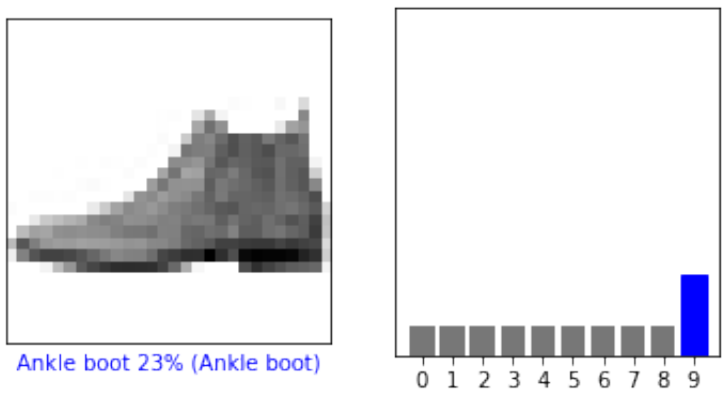
```
#membuat grafik
def plot_image(i, predictions_array, true_label, img):
    predictions_array, true_label, img = predictions_array[i], true_label[i], i
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(img, cmap=plt.cm.binary)

    predicted_label = np.argmax(predictions_array)
    if predicted_label == true_label:
        color = 'blue'
    else:
        color = 'red'
    class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sanc
    plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                        100*np.max(predictions_array),
                                        class_names[true_label]),
                                        color=color)

def plot_value_array(i, predictions_array, true_label):
    predictions_array, true_label = predictions_array[i], true_label[i]
    plt.grid(False)
    plt.xticks(range(10))
    plt.yticks([])
    thisplot = plt.bar(range(10), predictions_array, color="#777777")
    plt.ylim([0, 1])
    predicted_label = np.argmax(predictions_array)

    thisplot[predicted_label].set_color('red')
    thisplot[true_label].set_color('blue')

#membuat grafik
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions, test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions,  test_labels)
plt.show()
```



✓ 0s completed at 09:12